Mapping Real Time Applications on NoC Architecture with Hybrid Multi-objective Algorithm

AH.BENYAMINA[1] P.BOULET[2], A.AROUI[3], S.ELTAR[1], K.DELLAL[1]

[1] Université d'Oran - Département d'informatique Algérie
benyanabou@yahoo.fr
[2] INRIA Lille - Nord Europe, F-59650 Villeneuve d'Ascq, France
pierre.boulet@lifl.fr
[3] Centre des Techniques Spatiales (CTS), 1 Avenue de la palestine, BP 13, 31200 Arzew,Oran
arouikader@yahoo.fr

1 Introduction

As silicon [6] technology keeps scaling, it is becoming technically feasible to integrate entire and complex systems on the same silicon die. As result of this Systems on Chip (SOC) are inherently heterogeneous and therefore complex, they are often formed by multiple processors of different types (RISC, DSP; ASIC, for examples) with dedicated hardware or reconfiguration and peripheral.

Traditional concept of computer networking component interconnect-based routers, switches is adopted to extend the integration and perform the design. Network on chip (NOC) or plus generally MPSoCs are relatively aim new approach to integrated circuits on a platform SoC.[11]
Then MPSoCs and NoC are widely used in embedded systems (such as cellular phones, automotive control engines, etc..) where, once deployed in field, they always run the same set of applications.
In this paper we will focus on mesh-based NoC architectures, in which resources communicate with each other via mesh of switches that route and buffer messages. A ressource is generally any core: a general processor GP, a memory, an FPGA, DSP. A two dimensional mesh interconnection topology is the simplest from a layout perspective and the local interconnection between resources and switches are independent the network size.
Nevertheless, routing in a two dimensional mesh is easy, resulting in potentially small switches, high bandwidth, short clock cycles, and overall scalability [10]. One of the most onerous tasks in this context is the topological mapping of the resources on the mesh in such a way to optimize certain performances indexes (e.g power, performance). Mapping is, in fact, a problem of quadratic assignment that is known to be NP-hard.
The size space search of the problem increases exponentially with the system size depending on number of resources, tasks and communications. It is therefore of strategic importance to define methods to search a mapping that will optimize the desired performance indexes. In addition, the strategies have to handle a multi criteria exploration of the space of possible architectural mapping alternatives. The objectives to be optimized are, in fact, frequently multiple rather than single, and are almost always in contrast with each other. There is therefore non single solution to the problem of exploration (i.e single mapping) but a set of equivalent (i.e not dominated) possible architectural alternatives, featuring a different trade-off between the values of the objectives to be optimized (Pareto Set) [7].
Then a critical task for recent MPSoCs is the minimization of the energy consumed. We start from a well-characterized task graph, a directed acyclic graph representing a functional abstraction of the application that will run on the MPSoC. Each task is characterized by the number of clock cycles used for its execution. Clearly the duration of each task and the energy spent for running it depends on the clock frequency used during the task execution.
The problem we face is very complex. Because we solve, at the same time, the allocation of tasks to the processors and find the optimal path allocation (or communication mapping) referred in literature as Network Assignment [9].
For this, we have used two methods, one based on PSO algorithms and another on Dijkstra's algorithm. The solution must also verify some constraints such area, memory, load balancing, link speed, bandwidth and certainly hard real time constraints.

Our contribution differs from the above in that we use a new method in such research area and on the other hand, we try to map tasks on processors, and at the same time map optimally communications on the links of topology NoC.
The work presented in this paper is a contribution to solving a widespread problem in the field of system design, embedded the placement of a large application on an architecture (NOC).
Application is represented by a set of tasks that communicate with each other by sending message via bus on a heterogeneous architecture.
Our role is to place the tiles (task) on different elements (core) of architecture with the objectives of minimizing time execution and the energy consumption under the constraints of load balancing, bandwidth, available memory and size of the queue waiting processors [8].
To solve this problem, we used in the context of our present work, a new meta-heuristic algorithm Particle Swarm. it has proved its effectiveness in many fields such as optimization of networks, image processing and even control of industrial systems but it was never applied in our domaine.

Key words: Network-on-Chip, multiprocessor system-on-chip, multi-objective optimization, mapping, scheduling, genetic algorithms, Particle Swarm, Dijkstra.

2 Problem definition and Formulation

The communication between the cores of SoC is represented by core graph.
Definition 1 : The Task Graph (CG)is a directed graph, $G(V, E)$ with each vertex $V_i \in V$ representing a task and the directed edge (v_i, v_j), denoted as $e_{ij} \in E$, representing the communication between the tasks v_i and v_j. The weight of the edge e_{ij}, denoted by Q_{ij} represents the bandwidth of the communication from v_i to v_j. The connectivity and link bandwidth of the NoC is represented by the NoC topology graph.

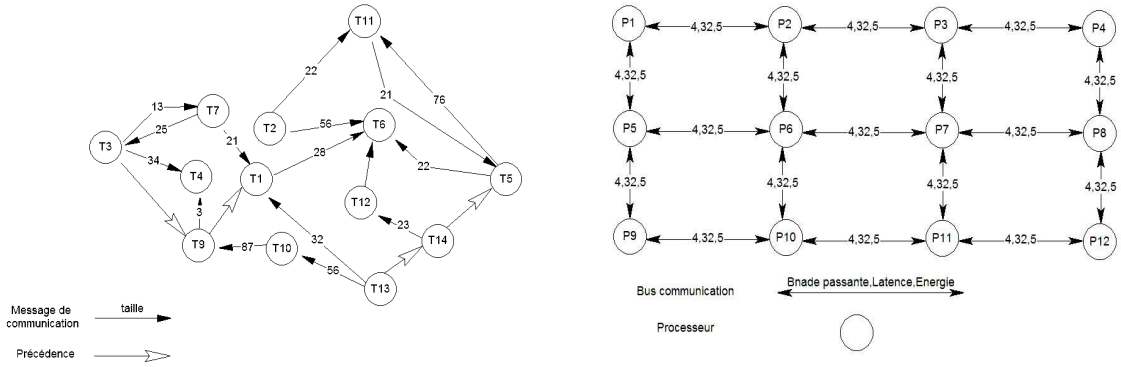

Fig. 1. Task graph and NoC graph

Definition 2 : The NOC Topology graph (NT) is a directed graph $P(U, F)$ with each vertex $u_i \in U$ representing a node in the topology and the directed edge (u_i,u_j) denoted as $f_{ij} \in F$representing a direct communication between the vertices's u_i and u_j. The weight of the edge f_{ij}, denoted by bw_{ij} represents the bandwidth available across the edge f_{ij} (see figure 1).

The mapping of the task graph $G(V, E)$ on to the processor graph $P(U, F)$ is defined by one to one mapping function map :
map : $V \rightarrow U$, s.t $map(v_i) = u_j \ \forall v_i \in V \ \exists u_j \in U$.
The mapping is defined when $\mid V \mid \geq \mid U \mid$ (see figure)

2.1 Mathematical formulation

For an CG each node represent one task with its characteristics or property.

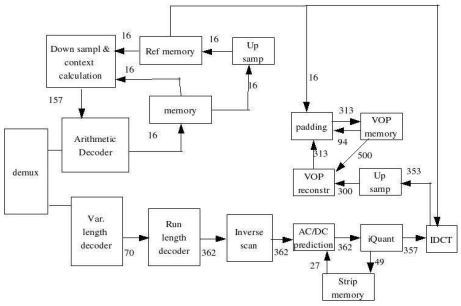

Fig. 2. block diagram of video object plane decoder with communication BW (in MB/s).

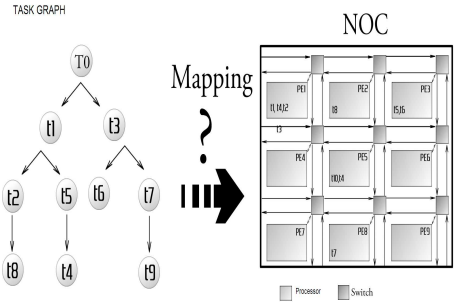

Fig. 3. Shown the mapping of task graph on NoC graph

Let $T = t_1, t_2, ..., t_n$ be the set of all tasks represented by CG.

$P = p_1, p_2, ..., p_s$ is the set of processors represented by the nodes in NT.

We consider that each processor p can run in different modes m_1, m_2 or m_3.

We model the allocation problem with binary variables X_{ij}^m such that [6]:

$X_{ij}^m = 1$ if $task_i$ is mapped on the processor j and runs in mode m, o otherwise.

d_{ij}^m = duration of execution task i on processor j running at mode m.

$d_{ij}^m = \frac{WCN_{ij}^m}{f_j^m}$ where WCN_{ij}^m is the number cycle needed by $task_i$ to be executed on processor j at mode m.

f_j^m is the frequency of the clock for processor j at mode m.

dl_i is deadline for $task_i$. The time at wish $task_i$ must be terminated.

There $dl_{final} = dl_n$ is deadline of last $task_n$ and of the application.

Q_{ij} is the size of data moved between $task_i$ and $task_j$.

dQ_{ijpq}^m is duration of communication between tasks i and j if they are assigned respectively to processor p and q at mode m (we see after how we compute dQ_{ijpq}^m).

q_{pq}^m is the duration of one unit (octet or bit) communication between p and q at mode m.

e_{pq}^m energy consumption for one unit from p to q at mode m.

If there is not a direct link between p and q let $\mu(p,q) = (pi, pj)$ be path form p to q.

Then duration using links is :

$dQ^m_{ijpq} = \sum Q_{ij} \times q^m_{plpk}$ where $(pl, pk) \in \mu(p,q)$ and $i \neq j$, $p \neq q$.
Let us note that if tasks i and j are mapped to the same processor the duration is negligible in comparison with a case where they are mapped to different processors. Therefore in the previous expression i differ from j and also p differ from q.
And consumption for communicating $task_i$ and $task_j$ if they are assigned to p and q at mode m towards the same path is :

$E^m_{ijpq} = \sum Q_{ij} \times e^m{}_{plpk}$ where $(pl, pk) \in \mu(p,q)$

Since we also take into account communication, we assume that two communicating tasks running on the same processor do not consume any energy and do not spend any time (indeed the communication time and energy spent are included in the execution time and energy), when they are allocated on two different processors, they both consume energy and spend time. Each path contains some switchs and routers that need power consumption and duration.[5]
Ascia and al [7] address this problem but not explain how compute this. In This work we will consider an average value of consumption (Csw) and duration (dsw); we can estimate theses considering communications, input and output to router as stochastic.

Then if $\mid \mu(p,q) \mid$ is the length of path $\mu(p,q)$, the total consumption of switch and router on this path is :

$\mid \mu(p,q) \mid +1) \times C_{sw}$

And the total duration is :

$\mid \mu(p,q) \mid +1) \times d_{sw}$

We can now do equation of the total consumption due to communication between $task_i$ and $task_j$ assigned respectively at processor p and q at mode m :

$$E^m_{ijpq} = E^m_{ijpq} + (\mid \mu(p,q) \mid +1) \times C_{sw} \tag{1}$$

and duration due to communication as :

$$dQ^m_{ijpq} = dQ^m_{ijpq} + (\mid \mu(p,q) \mid +1) \times d_{sw} \tag{2}$$

We explicit now total consumption and duration of processors.
For one $task_i$ mapped at $processor_p$ the duration is the sum of its start time and duration of all its communications with other tasks mapped to other processors. The strategies scheduling is LS with ASAP (As Soon As Possible). Then for the $task_i$ mapped on $processor_p$ its duration D_{ip} is done by the following equation :

$$D_{ip} = d^m_{ip} + dQ^m_{ijpq} + (\mid \mu(p,q) \mid +1) \times d_{sw} \text{with} i \neq j, p \neq q \tag{3}$$

$Dstart_i$ is the time at wish $task_i$ begin execution. It is equal at the time of the end of the last task which precedes it.
The duration of the application mapped on many processors it is equal at time end of the last task. Let D be total duration including time execution and communication over links of all tasks.

$$D = Dstart_n + D_n \tag{4}$$

Were D_n it is the duration of $task_n$ that is the last task. $Dstart_n$ is the time at wish $task_n$ begins execution.

The total consumption including processor, link and switch consumption is done by the flowed equation :

$$E_{pr} = \sum_{i=1}^{N} \sum_{p=i+1}^{s} \sum_{m=ml}^{mn} x_{ip}^{m} \times ET_{ip}^{m} \tag{5}$$

Where ET_{ip}^{m} is the consumption of $task_i$ if it is assigned on processor p at mode m.
Then E_{pr} is total energy consumed by all processors in NoC.

$$E_{com} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \sum_{p=1}^{s} \sum_{q=p+1}^{s} \sum_{m=ml}^{mn} x_{ip}^{m} \times x_{jp}^{m} \times E_{ijpq}^{m} \tag{6}$$

E_{com} is the total energy consumed by network (links, routers or switch)to assure all communications between all tasks overall the NoC.

2.2 Our multi-objective Model

Our method is based on evolutionary computing method and an optimizer path. We have to search a set of solutions under $multi-objective$. We consider here two objectives total duration and consumption.

First objective is duration D (4)
The second objective is the total power consumption done by E such that :

$$E = E_{pr} + E_{com} \tag{7}$$

Note date during computing D and E we look for the shortest path between two cores which satisfy the constraints (such bandwidth and buffer). To obtain this we used dijkstra. we refer to the problem as AAS (Assignation Affectation and Scheduling.)

3 AAS Problem resolution

The rationale behind our approach is the minimization of total power consumption in order to augment the autonomy of system. Nevertheless, trying to reach this objective increase time computing. Or embedded applications are generally real time and of course the deadline must not be exceed. Minimizing power consumption increases time computing and minimizing time computing increases power consumption. We have here two contradictory objectives, what returns this very complex problem. Multi-objective problems have a set of Pareto-optimal solutions. Each solution represents a different optimal trade-off between the objectives and is said "non-dominated" since it is not possible to improve one criterion without worsening another. We propose a multi-objective approach based on Particle Swarm optimization technique to solve our AAS problem.

3.1 PSO: Particle Swarm Optimization

A.Capone and M.Cesana proposed [3] an evolutionary population-based heuristic for optimization problems. It models the dynamic movement or behavior of the particles in a search space. By sharing information across the environment over generations, the search process is accelerated and is more likely to visit potential optimal or near-optimal solutions. PSO has been extended to cope with multi-objective problems which mainly consist of determining a local best and global best position of a particle in order to obtain a front of optimal solutions. One of the well-known multi-objective techniques based on PSO algorithm is MOPSO [4]. It is able to generate almost the best set of non dominated solutions close to the true Pareto front. The main algorithm is given below.

$$E^m_{ijpq} = E^m_{ijpq} + (\mid \mu(p,q) \mid +1) \times C_{sw} \tag{8}$$

Algorithm 1 MOPSO-MAIN

input : Swarm at iteration t S^t, MaxArchiveSize, MaxIteration
Output : Repository REP
*Step*0 :Initialization of Swarm
Initialize S at iteration $t = 0$
for each i $\in S^0$ **do**

  for each dimension d **do**
    Initialize $position_i$, save $pBest_i$, initialize velocity
    Specify $lowerbound_i$ and $upperbound_i$
  end for
end for
*Step*1 : Evaluation of particles S
*Step*2 : Update REP
for each i $\in S^t$ **do**
  compVector(i,REP)search-insert(S,REP)
end for
$STEP3$: Generate Mapping(associative grid): make-Cost(Mincost)
*Step*4 : Update Swarm :
for each i $\in S^t$ **do**

  for each dimension d **do**
    $Update - velocity_i$, $Update_position_i$
  end for
end for
*Step*5 : Boundary chek
*Step*6 : Update pBest
*Step*7 :
if $t > MaxIteration$ **then**
  Stop
end if
$t = t + 1$ and GO TO *Step*1

The following are the phases involved in the resolution of the proposed algorithm. In continuous optimization problems, getting the initial position and velocity is more straightforward because random initialization can be used. However, since the mapping problem is a constrained optimization problem, the initial positions must represent feasible solutions. Thus, they need to be designed carefully.[2]
A position in the search space represents a set of assignments that is a solution to the problem. In our case, each position provides information about how processor in the NoC will execute each task. Than, for each position in the swarm, we assign a Boolean value to the variables X^m_{ij}. We consider a feasible solution, a solution that satisfies all hard and soft constraints. During the search, only non-feasible solutions that violate some soft constraints can be included in the population. This increases the likelihood of a non-feasible solution to mutate and provide a feasible one in later generation.

3.2 Algorithm description

Given the rapidly changing and increasingly complex systems on chip (SoC - System on Chip) to systems on chip multi-processors (MPSoC - multiprocessor SoCs), interconnection of communication modules or cores (IP – Intellectual Property) constituting these systems, has undergone a change both in topology of the structure. This responds to the constraints of performance and

cost related to the complexity and the increasing of interconnected modules or IPs. Currently this process is moving towards the integration of a communication network on chip, implementing the transmission of data packets to nodes interconnected network corresponding to modules or IPs (processors, memory, peripheral controllers connected, etc..). This transmission is done through routers forming the network and implementing rules of referral and routing packets across the network.
In the design flow of an embedded system, the stage of investment and is directly related to the implementation of the application on an architecture specialist. The entries in this phase are:

- An application model
- A model of target architecture
- Constraints of performance and energy
- The objective functions to optimize

The output of this phase is an allocation of tasks and communications in natural resources, according to various tasks on these resources.

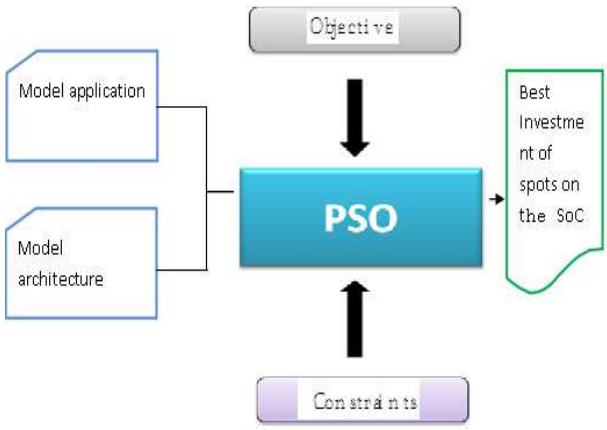

Fig. 4. PSO Algorithm

To run a distributed application, it is necessary to determine the best placement of spots that compose the target NoC architecture while reducing the execution time and energy constrained load balancing, memory, and the size of the queue. Our proposal for solving the problem of mapping is defined as follows:
The particle is a representation of the solution of the problem wich, in this case, describes the investment. If you have a NoC Mesh with S processors and an application with N tasks when the particle is a matrix of N line and S column.
As an example take 2 processors and 3 tasks, then the particle is the mapping of 3 tasks: $Tasks2$ and 3 are assigned to P_1, $task1$ is assigned to P_2.

P_1	P_2
0	1
1	0
1	0

Table 1. One Particle

Get a set of points describing the Pareto front :

– Estimated front by iterative algorithms generate points near the front
– eliminating dominated points
– Problems of convergence:

1. approach the front
2. cover the entire front
3. Concept archive: Keep each iteration all the points not dominated.

Then the problem of placement of tasks on a NoC is to minimize cost objective function. It can be formulated as follows:

– Given the graph of application (size of the task type of Soc, runtime memory required by processor bandwidth required for a message and message size).
– Given the architectural graph (speed performance by mode, power consumption by mode, load balancing (load minimum and maximum), available memory in processor, size of the queue and bus latency and energy consumption due to transmission
– From the placement of tasks on the processor's by different modes. This is equivalent to Minimize F (X): (f (time) f (energy)) The determination of the fitness function (or adaptive function evaluation) involves several steps. Each time a swarm is generated according to the fitness of each particle must to be evaluated.
– A particle represents a distribution of tasks of the application in the target NoC architecture.
– For a particle, move the communication costs of all messages for each message eliminating paths wich bandwidth is less than that required by the message (using the algorithm of the shortest path), then: We calculate the execution time of each task by mode which has been allocated within the processor.

3.3 Optimizing communication and energy through the Network

To minimize communication and power consumption another approach is necessary to find the optimal objectives. We have used a method based on dijkstra algorithm. Then, we crosse this method with MoPSO, described previously to approach our global objectives.

Algorithm for optimizing path rooting

1– Begin
2– Read identifiers (processor origin Po, processor target Pt)
3– Read matrix bandwidth MB
4– Read matrix communication between tasks MC
5– Call dijkstra (Po,Pc,MB,MC)
6– Return set of links with optimal cost i.E Paths
7– Call verification (verify if bandwidth of all links still verified)
8– If constraint not verified GO 4 for another path
9– Return optimal path
10– END

The Pareto Front is generated from existing solutions in the archive.

4 Experimentations and results analysis

The algorithm is coded in JAVA programming language and all the experiments were carried out on a Pentium 3.2 GHz.

We have varied the number of generations for a same example to know what is the best interval of generations is for similar size of applications. In general study the searchers fixe this number at 20 and our experiments shows that not necessarily to take this number bigger than 20 nevertheless our example is not taken with a big size(5).

The other parameter of the algorithm wish is the size of swarm is also important because he influence the convergence of research. Our experiment shows that it is important to take Swarm size near 100.(5)

We study the performance of our algorithm over many examples and the results are shown in the following figure. We also compared it with the results obtained with the genetic algorithm having been the object of previous works of our part[5].

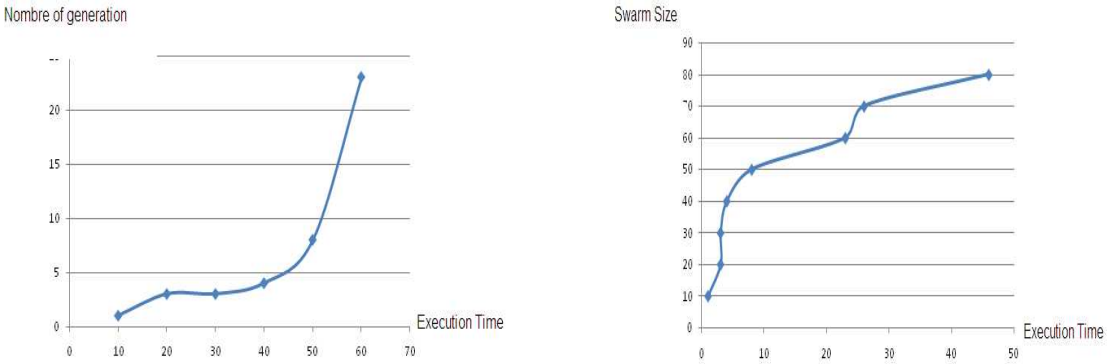

Fig. 5. influence of number generation on Time computing and influence of size swarm on Time computing

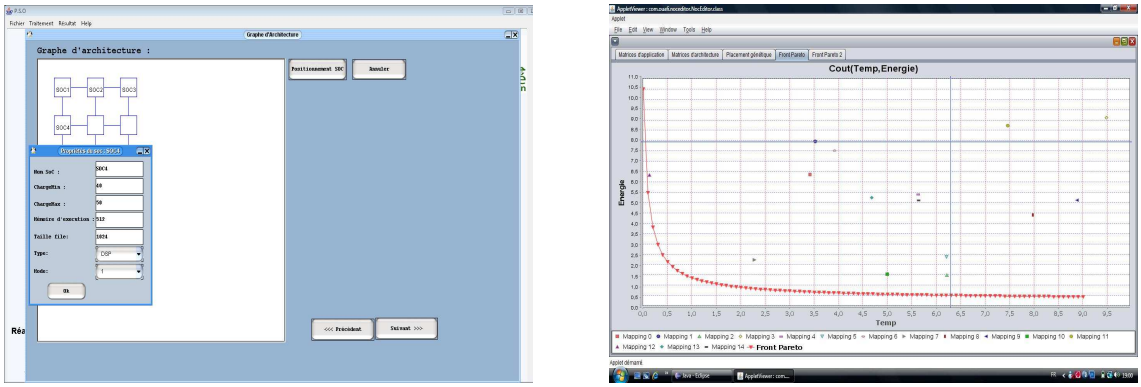

Fig. 6. Application interface and Feasible solutions generated with the same initial parameters

4.1 Comparison between PSO and GA (genetic algorithm)

Genetic algorithms belong to the class of evolutionary algorithms which are themselves a branch of meta heuristics-based populations, such as methods of particle swarm optimization. All these meta heuristics have in common is that they are all inspired from biological phenomena.

Generally most of these evolutionary techniques have the following steps:

1. Random generation of initial population.
2. Calculation of the fitness value for each subject.
3. Reproduction of the population based on fitness values.
4. If needs are met, then stop. Otherwise return to step.

In this process, we can learn that PSO shares many common points with the genetic algorithm. On the one hand, PSO does not use the genetic operators such as crossing and mutation. The particles are updated with the internal speed. They also have memory, which is important for the algorithm (although this memory is very simple because it only stores the positions pBest and gBest). In addition, the mechanism for sharing information in PSO is different: In GA, the information is shared among the chromosomes to each other. In PSO, only gBest (the value of fitness of the best known of the neighborhood) provides information to other particles to follow the leader. By comparing the runtime complexity of both algorithms, we should exclude similar operations (initialization, fitness evaluation, and stop running). We also exclude the number of generations, because it depends on stopping criteria and complexity of the problem of optimization (PSO experiments show that needs a number of generations a little higher than GA to achieve a quality given solution). Therefore, we focus our comparison on the main loop of the two algorithms. We consider the longest process (recombination in GA and that the update speed and position in PSO). The complexity of GA is greater than that of PSO. Therefore, PSO is sampler and faster than GA.

	$NbPr$	$NBTask$	$executiontime$	$Energy$
PSO	2	4	348	14053
PSO	4	6	3152	24025
PSO	6	8	13659	302430
GA	2	4	164.61	8560
GA	4	6	1718	20619
GA	6	8	75985	29693

Table 2. Comparison between PSO and GA

We have compared the two methods on examples with average size. The results are given by the following table :

This table summarizes a study that is done in order to compare the results of both PSO and GA approaches. This study was conducted to test a set of algorithms of both PSO and GA approaches to the problem of applications mapping on architectures (MPSoC) and analyzing their output results. Table shows that the results given by GA are better than those given out by PSO.

5 Conclusion

Particle swarm optimization (PSO) is a heuristic search algorithm which is a relatively new heuristic based on collaboration and swarming in biological populations. Our approach presented in this article, for the problem of applications scheduling on MPSoC architectures, is based on the PSO method, hybridizing with the shortest path algorithm Dijkstra's to improve performance and achieve desired goals.

PSO is similar to the genetic algorithm (GA) in the sense that they are two research approaches based on population and they both depend on the sharing of information among their members of the population to improve their research process using a combination of deterministic and probabilistic rules. It was noted at the end of this article, as all test scenarios conducted to compare the two approaches shown that the genetic approach is better than PSO approach in terms of efficiency and accuracy of results.

Then we consider for the moment this work a first step in our search for a good meta heuristic that can be crossed with other exact methods to solve the overall problem known as GILR.

References

1. IR.Quadri, P.Boulet, S.Meftali and JL.Dekeyser, *Using an MDE Approach for Modeling of Interconnection Networks*, Proceedings of the The International Symposium on Parallel Architectures, Algorithms, and Networks. ISPAN, pp 289-294, 2008.
2. HJ.Escalante, M.Montes and LE.Sucar *Particle Swarm Model Selection*, The Journal of Machine Learning Research. pp.405-440, vol 10, 2009.
3. A.Capone, M.Cesana, I.Filippini and F.Malucelli, *Optimization models and methods for planning wireless mesh networks* , Computer Network. accepted for publication January 2008.
4. TDL.Truong, *Shared protection for multi-domain networks*, Ph.D.thesis. University of Montreal. Sept.2007
5. AH.Benyamina, P.Boulet and B.Beldjilali , *An Hybrid algorithm for Mapping on NoC Architectures*, META'08 . Hammamat,Tunisia,October, 2008.
6. A.Benymina and P.Boulet,*Multi-objective Mapping for NoC Architectures* , Journal of Digital Information Management (JDIM). pages 378-384, volume 5, number 6, 2007.
7. G.Ascia, V.Catania and M.Palesi, *A Multi-objective Genetic Approach to Mapping Problem on Network-on-Chip* Journal of Universal Computer Science. pages 370-394, vol 2, (2006)
8. AH.Benyamina, B.Beldjilali, S.Eltar, K.Dellal, *Mapping Real Time Applications on NoC Architecture with Hybrid Multi-objective PSO Algorithm*,COSI'2010 . Ouaragla, Algérie, Avril, 2010.
9. D.Shin, J.Kim,*Power-Aware Communication Optimisation for Networks-on-Chips with Voltage Scalable Links* , in Proc. CODES+ISSS'04, pp.170-175, Sep.2004.
10. L.Benini, D.Bertozzi, A.Guerri and M.Milano, *Allocation, Scheduling and Voltage Scaling on Energy Aware MPSoCs*, CPAIOR pp.44-58, 2006.
11. T.Bjerregaard and S.Mahadevan, *survey of research and pratices of network-on-chip*, ACM Comput Surv. 38(1):1, 2006.